 GLAST LAT SPECIFICATION	Document # **LAT-DS-00455-01**	Date Effective 29 November 2001
	Prepared by(s) Daniel Wood	Supersedes None
	Subsystem/Office Calorimeter Subsystem	
Document Title **Calorimeter Pre-Electronics Module Software Design Description**		

Gamma-ray Large Area Space Telescope (GLAST)

Large Area Telescope (LAT)

Calorimeter Pre-Electronics Module Software Design Description

(V0-0-1)

1. OVERVIEW 3

2. COMMANDS 4

COMMAND OPCODES 5
SYS COMMANDS 6
ADC COMMANDS 7
CFD COMMANDS 8

3. EVENT DATA 9

4. RUNNING THE SERVER 11

5. A UNIX COMMAND CLIENT 12

CONNECT 13
DISCONNECT 13
START 13
STOP 14
ENABLE 14
DISABLE 14
THRESHOLD 15
PEDESTAL 15
TRIGGER 15
PULSE 16
HELP 16
EXIT 16

6. A UNIX EVENT CLIENT 16

1. Overview

The CAL Pre-Electronics Module (PEM) server is a TCP/IP server which provides a low-level interface to the CAL PEM hardware. The PEM interface hardware consists of a set of VME digital and analog modules. The PEM server handles the interface to the following modules:

C.A.E.N. V785 Peak Sensing Converter

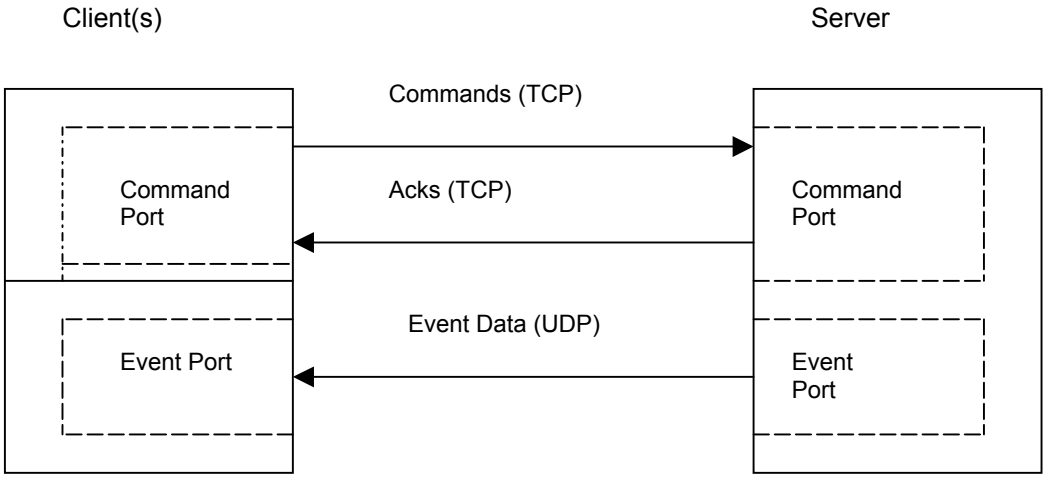
C.A.E.N. V775 Time-to-Digital Converter

C.A.E.N. V812 Constant Fraction Discriminator

The Peak Sensing Converter and Time-to-Digital Converter modules convert analog values into digital event data. These ADC modules also support masking and zero-suppression of channels. The Constant Fraction Discriminator threshold values may be programmed.

The PEM server is designed to run on a VME single-board computer (SBC) using the VxWorks operating system. The SBC serves as the VME system controller for all of the modules. The PEM server software writes to the VME modules for configuration and reads from the VME Modules to obtain event data. The server listens for TCP client connections on the command port. The server sends out event data on the event port as UDP datagrams. The port number values and the event client IP address are specified as startup parameters to the server. The schematic below shows the basic data flow:

Figure 1 - CAL PEM Data Paths



The PEM client may be any machine and application that supports standard IP operation. The command client and event client processes may be different applications and may run on different machines.

The PEM server expects that the hardware trigger signals are common to all of the Peak Sensing Converter and Time-to-Digital Converter boards. That is, a GATE or COM signal to

one board should be broadcast to all other boards. This restriction limits the difficulties in recovering event data from multiple boards triggering at different rates. The PEM server, however, does support variable length event data for each trigger. This is accomplished by configuring the zero-suppression thresholds in the ADC modules.

2. Commands

The CAL PEM server accepts a number of low-level commands to configure the setup. Each command arrives on the TCP command connection as a 16 byte packet. The format of the command packet is shown below:

Figure 2 - PEM Command Packet

Sync	
Sequence	
Opcode	Board
Channel	Parameter

Sync – A fixed word to indicate the beginning of the next command (32 bits) = 00000001 hex

Sequence – A counter which increments by one for each command sent (32 bits)

Opcode – The command action code (16 bits)

Board – The target board number (16 bits)

Channel – The target channel number (16 bits)

Parameter – Extra information dependent on the *Opcode* value (16 bits)

Each command received by the CAL PEM server will result in the server sending an acknowledge packet back to the client. Each acknowledge packet is sent back on the TCP command connection as a 16 byte packet. The format of the command packet is shown below:

Figure 3 - PEM Acknowledge Packet

Sync	
Sequence	
Opcode	Board
Channel	Status

Sync – A fixed word to indicate the beginning of the next acknowledgement (32 bits) = 00000002 hex

Sequence – This value matches the *Sequence* count for the command packet, allowing acknowledgments to be matched with commands (32 bits)

Opcode – The *Opcode* value from the command packet (16 bits)

Board – The *Board* value from the command packet (16 bits)

Channel – The *Channel* value from the command packet (16 bits)

Status – The completion code of the command dependent on the *Opcode* value (16 bits)

Command Opcodes

Each command opcode is part of a module group. The following module groups are supported:

Table 1 - PEM Module Groups

Module Group	Description
SYS	System
ADC	C.A.E.N V785 Peak Sensing Converter (PSC) C.A.E.N. V775 Time-to-Digital Converter (TDC)
CFD	C.A.E.N V812 Constant Fraction Discriminator

The SYS module represents the whole system for those commands that require coordinated activities among multiple modules. The PSC and TDC modules present an identical interface to the server software, so these modules are grouped together. The distinguishing feature will then be the module board number.

The list of known opcodes is shown below:

Table 2 – PEM Command Opcodes

Opcode	Module	Command
1	SYS	Start Acquisition
2	SYS	Stop Acquisition
5	SYS	Test Trigger
6	SYS	Test Pulse
11	ADC	Enable Channel
12	ADC	Disable Channel
13	ADC	Set Threshold
14	ADC	Set Pedestal
21	CFD	Enable Channel
22	CFD	Disable Channel
23	CFD	Set Threshold

SYS Commands

SYS Start Acquisition

Enables event data for writing to the Event Data Socket.

Opcode: 1
Board: Indicates which ADC board will generate the event data ready interrupt.
Channel: ignored
Parameter: ignored
Status: 0000 = success, FFFF = failure

SYS Stop Acquisition

Disables event data for writing to the Event Data Socket.

Opcode: 2
Board: ignored
Channel: ignored
Parameter: ignored
Status: 0000 = success, FFFF = failure

SYS Test Trigger

Sends a software test trigger signal to the ADC boards.

Opcode: 5
Board: ignored
Channel: ignored
Parameter: The number of test triggers to send. Each trigger is spaced one system clock tick apart on the server machine (nominally 10 ms)
Status: 0000 = success, FFFF = failure

SYS Test Pulse

Sends a test output pulse on all of the CFD boards on all channels.

Opcode: 6
Board: ignored
Parameter: The number of test pulses to send. Each pulse is spaced one system clock tick apart on the server machine (nominally 10 ms)
Status: 0000 = success, FFFF = failure

ADC Commands

ADC Enable Channel

Enables a channel's data for event readout.

Opcode: 11
Board: board number (0 – 31, or FFFF for all boards)
Channel: channel number (0 – 31, or FFFF for all channels)
Parameter: ignored.
Status: 0000 = success, FFFF = failure

ADC Disable Channel

Disables a channel's data for event readout.

Opcode: 12
Board: board number (0 – 31, or FFFF for all boards)
Channel: channel number (0 – 31, or FFFF for all channels)
Parameter: ignored.
Status: 0000 = success, FFFF = failure

ADC Set Threshold

Sets the zero-suppression threshold for the given channel.

Opcode: 13
Board: board number (0 – 31, or FFFF for all boards)
Channel: channel number (0 – 31, or FFFF for all channels)
Parameter: threshold value (0 – 255)
Status: threshold value, or FFFF = failure, or 0000 = success for multiple boards or channels

ADC Set Pedestal

Sets the pedestal value for the given channel.

Opcode: 14
Board: board number (0 – 31, or FFFF for all boards)
Channel: ignored
Parameter: pedestal value (0 – 255), or FFFF = use sliding scale feature
Status: pedestal value, or FFFF = failure, or 0000 = success for multiple boards

CFD Commands

CFD Enable Channel

Enables a CFD channel.

Opcode: 21
Board: board number (0 – 31, or FFFF for all boards)
Channel: channel number (0 – 15, or FFFF for all channels)
Parameter: ignored
Status: 0000 = success, FFFF = failure

CFD Disable Channel

Disables a CFD channel.

Opcode: 22
Board: board number (0 – 31, or FFFF for all boards)
Channel: channel number (0 – 15, or FFFF for all channels)
Parameter: ignored
Status: 0000 = success, FFFF = failure

CFD Set Channel Threshold

Sets the discriminator threshold value for the given channel.

Opcode: 23
Board: board number (0 – 31, or FFFF for all boards)
Channel: channel number (0 – 15, or FFFF for all channels)
Parameter: threshold value (1 – 255)
Status: threshold value, or FFFF = failure, or 0000 = success for multiple boards or channels

3. Event Data

When event data acquisition is active, the CAL PEM server will send event data on the UDP event data socket. Each event corresponds to a set of synchronized PSC GATE and TDC COM signals. Each event arrives on the client event data socket as a single UDP packet. Each event (corresponding to one global trigger signal) contains one event header and multiple event data groups. The number of event data groups depends on the number of active ADC boards installed in the system. Each event data group contains the channel readout data from a particular board. The number of channels included in a board data group depends on the number of enabled channels on that board as well as whether or not the zero suppression feature is enabled for that board. The format of the 16 byte event header is shown below:

Figure 4 - PEM Event Packet Header

Number of Boards
Sequence
Time Stamp
Event List Length

Number of Boards – The number of ADC boards and event data groups contained in the packet (32 bits)

Sequence – A counter which increments by one for each event sent (32 bits)

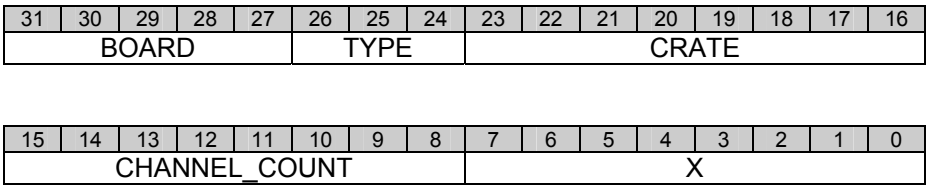
Time Stamp – A time value (32 bits)

Event List Length – The number of event data bytes remaining in the packet (32 bits)

The remaining event list data is a sequence of 32-bit words whose format is detailed in the *C.A.E.N Model V775 User's Manual* and *C.A.E.N Model V785 User's Manual*. Each event data group is tagged with a board number, channel count, and event count. The event data

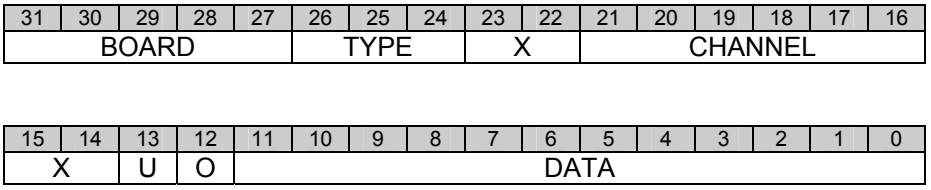
will be read from the ADC boards so that all of the channels for a given board are packaged together. Each event data group contains three types of data words (each 32 bits): a Header Word, multiple Data Channel Words, and a End-of-Block Word.

Figure 5 - Event Data Header Word



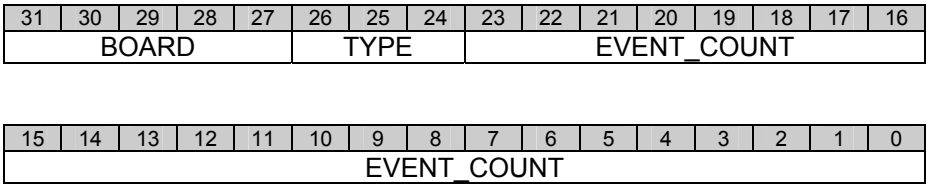
BOARD: The board number
TYPE: The data word type = 2
CRATE: The system crate number
CHANNEL_COUNT: The number of channel data words in this group
X: ignored

Figure 6 - Event Data Channel Word



BOARD: The board number
TYPE: The data word type = 0
CHANNEL: The channel number
U: Data underflow flag
O: Data overflow flag
DATA: The ADC value for this channel
X: ignored

Figure 7 - Event Data End-of-Block Word



BOARD: The board number
TYPE: The data word type = 4
EVENT_COUNT: The hardware event count. This value starts at '0' for every *Start* command or reset.

4. Running the Server

The *pemserver* application is implemented as three CMX VxWorks constituents in project *CAL*, package *PEM*. The server application itself is called *pemserver.exe*. The interfaces to the CAEN ADC and CFD boards are implemented in two shareables called *libcaen_adc.o* and *libcaen_cfd.o* respectively. The server application also requires the services of two other CMX constituent libraries: *libmv2x_extra.o* in project *BFS*, package *MV2X* and *libbbc.o* in project *BFS*, package *BBC*.

The server application may be started from the VxWorks shell by calling the entry point function *pemserver()* and passing in the startup parameters. The *perserver()* function takes 10 arguments:

```
pemserver <evt_addr>, <cmd_port>, <evt_port>, <adc_addr>,
<cfd_addr>, <intr_level>, <intr_vector>, <evt_thresh>,
<maj_thresh>, <pulse_width>
```

evt_addr - The IP address of the machine to receive the UDP event data packets. It may either be a registered host name string or a numerical dot address string.
cmd_port - The TCP port number to listen on for connections from the command client machine.
evt_port - The UDP port number to send the event data packet to.
adc_addr - The ADC board VME bus address block base.
cfd_addr - The CFD board VME bus address block base.
intr_level - The VME hardware interrupt level to use for the ADC event data ready interrupt. Valid values are 1 - 7.
intr_vector - The system interrupt vector number to use for the ADC event data ready interrupt. Valid values are 96 - 255.
evt_thresh - The event threshold level for the ADC board event data ready interrupt. The ADC boards will buffer this many events internally before issuing the interrupt. Valid values are 1 - 32.
maj_thresh - The majority threshold value to use for the CFD boards. Valid values are 6 - 244.

pulse_width - The output pulse width value to use for the CFD boards. valid values are 0 - 255.

The sample VxWorks shell script below shows the minimum needed to start the server.

Example 1 - PEM Server Startup Script

```
ld < /glast/flight/BFS/binary/BBC/V0-0-8/VxWorks/libbbc.o
ld < /glast/flight/BFS/binary/MV2X/V0-1-1/MV2300/libmv2x_extra.o

ld < /glast/flight/CAL/binary/PEM/V0-0-1/VxWorks/libcaen_adc.o
ld < /glast/flight/CAL/binary/PEM/V0-0-1/VxWorks/libcaen_cfd.o
ld < /glast/flight/CAL/binary/PEM/V0-0-1/VxWorks/pemserver.exe

pemserver "132.250.161.58", 9000, 9001, 0x8000000, 0x8200000, 7,
200, 16, 194, 255
```

Each VME bus electronics module group is assigned a 2 MB address window. Each board will then occupy a 64 KB slot within the window. The *adc_addr* and *cfd_addr* server startup parameters specify the base addresses of the two module groups. In the above example, for instance, there could be two ADC boards installed at addresses 0x80000000 and 0x80010000. At startup, the server software will probe the allowed slot bus addresses for each module group. The software will automatically detect the number of boards in each module group and number them by increasing address. The ADC boards in the example would be designated board 0 at 0x80000000 and board 1 at 0x80010000. The server will support a maximum of 32 boards in each module group. The board with the lowest VME bus address will always be designated as the lowest command address.

5. A UNIX Command Client

The *pemcmd* application is a TCP/IP client which provides a simple command and acknowledge interface to the PEM server. The *pemcmd* client is a command line application. It resides as a CMX constituent in project *CAL*, package *PEM*. The *pemcmd* client will run under both Solaris and Linux.

To start the command client, log in to the machine which will run the application. In a terminal window, type the following lines:

Example 2 - Running the PEM Command Client

```
machine%  cmx login
machine%  pemcmd
PEM>
```

The command client is now ready to accept commands. If the PEM server is not up and running, then it is still possible to issue dummy commands just to explore the syntax and check the command packet formatting. In this test mode, no command acknowledges are

present. Otherwise if the server is running, the *connect* command must be the first issued in order to actually send commands to the server machine.

The following commands are recognized by the *pemcmd* application:

Connect

Syntax

```
connect  <server_ip>  <server_port>
```

Description

Connects the command client to the command server. The server_ip parameter is the IP address of the server machine given either as a recognized host name or as a dot format numerical address. The *server_port* parameter is the TCPU port number at which the server listens for command connections and should match the startup parameter given to the server.

Disconnect

Syntax

```
disconnect
```

Description

Disconnects the command client from the server.

Start

Syntax

```
start  [--board=<n>]
```

Description

Starts the event acquisition. The optional *board* qualifier specifies which ADC board will generate the event data ready interrupt (default is 0).

Stop

Syntax

```
stop
```

Description

Stops the event data acquisition.

Enable

Syntax

```
enable  adc | cfd  [--board=<n>]  [ --channel=<n>]
```

Description

Enables an ADC or CFD channel. The optional *board* qualifier specifies which board to change (default is all boards). The optional *channel* qualifier specifies which channel to change (default is all channels).

Disable

Syntax

```
disable  adc | cfd  [--board=<n>]  [--channel=<n>]
```

Description

Disables an ADC or CFD channel. The optional *board* qualifier specifies which board to change (default is all boards). The optional *channel* qualifier specifies which channel to change (default is all channels).

Threshold

Syntax

```
threshold adc | cfd <thresh_value> [--board=<n>]
[--channel=<n>]
```

Description

For ADC boards, this command sets the zero suppression threshold for a channel. For CFD boards, this command specifies the actual discriminator threshold. The optional *board* qualifier specifies which board to change (default is all boards). The optional *channel* qualifier specifies which channel to change (default is all channels).

Pedestal

Syntax

```
pedestal <ped_value> [--board=<n>]
```

Description

Specifies the ADC pedestal value for a board. The optional *board* qualifier specifies which board to change (default is all boards).

Trigger

Syntax

```
trigger [num_trig]
```

Description

Sends one or more test triggers for all of the ADC boards. The optional *num_trig* parameter specifies the number of triggers (default is 1).

Pulse

Syntax

```
pulse [num_pulse]
```

Description

Sends one or more test pulses for all of the CFD boards. The optional *num_pulse* parameter specifies the number of pulses (default is 1).

Help

Syntax

```
help
```

Description

Prints a summary of the client commands.

Exit

Syntax

```
exit
```

Description

Quits the client application

6. A Unix Event Client

The *pemdump* application is a UDP/IP client which provides a simple event data interface to the PEM server. It resides as a CMX constituent in project *CAL*, package *PEM*. The *pemdump* client will run under both Solaris and Linux.

To start the event client, log in to the machine which will run the application. In a terminal window, type the following lines:

Example 3 - Running the PEM Event Client

```
machine% cd my_data_dir
machine% cmx login
machine% pemdump 9001
```

In this example, 9001 is the UDP port number to which the PEM server will send event data. This should match the value given as a startup parameter to the server. Also, the server should have been started with the parameter giving the IP address of the machine running the *pemdump* application.

The instructions above start the *pemdump* application in record-only mode. The application will always open a new event data file in the directory from which it was launched. This is the reason the "`cd my_data_dir`" command is issued before starting. The *my_data_dir* path is any user directory for storing the event data files. The *pemdump* application always automatically generates the event data file name. It is of the format: `<unix_time>.cpd`, where *unix_time* is the current UNIX time. The UNIX time is the number of seconds since Jan 1, 1970 00:00:00 GMT. This convention ensures that the data file name are unique and that they are always ordered by time of capture. The application will print the file name it has chosen at startup.

The *pemdump* application supports an optional extended mode of operation:

```
machine% pemdump 9001 -v
```

The *–v* flag indicates that the application should display the event data contents as they are received. Here is an example of one event display in the verbose mode:

Example 4 - PEM Event Client Display

```
[0000000f] Boards: 1, Time: 8608, Length: 136

Crate: 0   Board: 2   Channels: 32
Chan 00 = -7
Chan 16 = 7
Chan 01 = 1
Chan 17 = 1
Chan 02 = -4
Chan 18 = 7
Chan 03 = 0
Chan 19 = 6
Chan 04 = -3
Chan 20 = 1
Chan 05 = 0
Chan 21 = 5
Chan 06 = 2
Chan 22 = 5
Chan 07 = -3
Chan 23 = 5
Chan 08 = -8
Chan 24 = 2
Chan 09 = 2
Chan 25 = 10
Chan 10 = 0
```

```
Chan 26 = 6
Chan 11 = 2
Chan 27 = 2
Chan 12 = 3
Chan 28 = 0
Chan 13 = 0
Chan 29 = 4
Chan 14 = 0
Chan 30 = 4
Chan 15 = -2
Chan 31 = 4
Event Count: 15
```

The first line gives the event packet software sequence count in hex followed by the timestamp and size of event data in bytes. The next line gives the VME crate number, hardware ADC board number, and the number of channels above threshold. The majority of the display shows the raw ADC value for each channel over threshold. The final line gives the ADC hardware event count. The *pemdump* application records data as normal in verbose mode. Event rates should be limited in verbose mode to avoid flooding the display with too much text.